CLAIMS

1. A method in a computer system for accessing objects of varying thread types from a program written in a language that does not support objects of varying thread types, the thread types including thread-safe and not thread-safe objects, the method comprising:

creating a plurality of threads within a process, one thread being designated as an apartment thread, the apartment thread for receiving messages to instantiate objects and to invoke methods of the instantiated objects;

when the program designates to instantiate an object,

determining the thread type of the object;

when the thread type is not thread-safe, sending a message to the appropriate thread to instantiate the object; and

when the thread type is thread-safe, instantiating the object within the currently executing thread; and

when the program designates to invoke a method of the instantiated object,

determining the thread type of the object;

when the thread type is not thread-safe, sending a message to the appropriate thread to invoke the method of the instantiated object; and

when the thread type is thread-safe, invoking the object within the currently executing thread.

2. The method of claim 1 wherein the creating of the thread designated as the apartment thread is deferred until an apartment-threaded object is to be instantiated.

3. The method of claim 1 wherein the thread type includes a main-threaded wherein one of the plurality of threads is designated as a main thread, and

wherein main-threaded objects are instantiated by the main thread and method of main-threaded objects are invoked by the main thread.

4. The method of claim 1 wherein the programming language is a Java-based language and the objects adhere to the Microsoft Component Object Model.

5. The method of claim 1 wherein when the program designates to instantiate an object, a wrapper object is instantiated by the current thread, the wrapper object containing a reference to the instantiated object and an indication of whether the object is thread-safe or not thread-safe.

6. The method of claim 5 wherein the wrapper object contains the identification of the apartment thread when the object is apartment-threaded.

7. The method of claim 5 wherein the wrapper object contains a null identification of a thread when the object is thread-safe, the null identification indicating that the object can be accessed by any thread.

8. The method of claim 1 wherein when an invoked method returns a reference to an object, storing the identification of the returning thread as the thread to access the referenced object.

9. The method of claim 1 wherein when an invoked method returns a reference to an object,

saving an indication of the returning thread from which the method was invoked; and

when invoking a method of the referenced object,

requesting the returning thread to marshal another reference to the referenced object;

determining whether the marshaled reference is the same as the returned reference;

when the marshaled reference is the same as the returned reference, invoking the method from the current thread; and

when the marshaled reference is not the same as the returned reference, sending a message to the returning thread to invoke the method of the referenced object.

10. A method in a computer system for determining whether a method of an object can be invoked from the current thread, the object being instantiated by an instantiating thread and being identified by a first reference, the method comprising:

requesting the instantiating thread to marshal a second reference to the object to the current thread;

when the second reference is received,

determining whether the second reference is the same as the first reference;

when the references are the same, invoking the method of the object from the current thread; and

when the references are not the same, requesting that the instantiating thread invoke the method of the object.

11. The method of claim 10 including saving an indication of whether the instantiating thread should be used to invoke methods of the object so that the instantiating thread is requested to invoke the method of the object for subsequent invocations of methods of the object.

12. The method of claim 10 including saving an indication of whether any current thread can be used to invoke the method of the object so that any current thread can invoke methods of the object.

13. The method of claim 10 wherein the instantiating thread uses a marshaling member functions of the object to marshal the second reference and wherein when the object is thread-safe, the marshaling member function marshals a pointer that points directly to the object.

14. The method of claim 10 wherein object is developed to adhere to the Microsoft Component Object Model.

15. A method in a computer system for accessing Microsoft Component Object Model ("COM") objects of varying thread types from a Java-based program, the computer system having a Java virtual machine ("VM") for executing statements of the Java program, the Java VM executing in a multithreaded process, the method comprising:

when executing a statement of the Java-based program to instantiate a COM object,

instantiating a wrapper object;

when the thread type of the COM object is not thread-safe,

requesting that the appropriate thread instantiate the COM object; and

storing an identifier of the appropriate thread in the wrapper object;

when the thread type of the COM object is thread-safe,

instantiating the COM object from the current thread; and

storing an indication in the wrapper object that the COM object can be accessed from any thread;

storing a reference to the instantiated COM object in the wrapper object; and

using a pointer to the wrapper object as a reference to the COM object; and

when executing a statement of the Java-based program to invoke a member function of the COM object referenced by the pointer to the wrapper object,

when the wrapper object contains an identifier of a thread, requesting the identified thread to invoke the member function of the COM object referenced by the wrapper object; and

when the wrapper object indicates that the COM object can be referenced from any thread, invoking the member function of the COM object referenced by the wrapper object directly from the current thread.

16. A method in a computer system for mapping access to Microsoft Component Object Model ("COM") objects to a Java programming model, the computer system having a Java virtual machine ("VM") for executing statements of the Java program, the method comprising:

when executing a statement of the Java-based program to instantiate a COM object,

instantiating a wrapper object;

instantiating the COM object; and

storing a reference to the instantiated COM object in the instantiated wrapper object;

using a pointer to the wrapper object as a reference to the COM object; and

when executing a statement of the Java-based program to invoke a member function of the COM object referenced by the pointer to the wrapper object,

invoking the member function of the COM object;

when the member function returns an indication of an error, generating an exception; and

setting a return value of the member function to a return parameter of the member function.

17. The method of claim 16 wherein a class definition file for the COM object contains an indication of which parameter of the member function of the COM object should be returned as the return value of the member function according to the Java programming model.

18. The method of claim 16 wherein when the COM object is apartment-threaded, requesting an apartment thread to access the COM object.

19. A method in a computer system for mapping access to an object developed with a first programming model to a second programming model, the method comprising:

intercepting an attempt to instantiate the object;

instantiating the object; and

setting a wrapper object to reference the instantiated object;

intercepting an attempt to invoke a member function of the instantiated object using a reference to the wrapper object;

invoking the member function of the object referenced by the wrapper object; and

mapping parameters returned by the invoked member function in accordance with the first programming model to the second programming model.

20. The method of claim 19 wherein the first programming model is the Microsoft Component Object Model.

21. The method of claim 20 wherein the second programming model is a Java programming language model.

22. The method of claim 19 wherein the mapping of parameters includes when a result status that indicates an error, is returned by the member function generating an exception.

23. The method of claim 19 wherein the mapping of parameters includes setting a return value of the member function to a parameter returned by the member function.

24. A method in a computer system for accessing objects of varying thread types in a manner that is transparent to a program accessing the objects, the method comprising:

when the program indicates to instantiate an object,

when the thread type of the object indicates that it is accessible only by a certain thread, requesting that certain thread to instantiate the object; and

when the thread type of the object indicates that it is accessible by any thread, instantiating the object from the current thread; and

when the program indicates to access the object,

when the thread type of the object indicates that it is accessible only by a certain thread, requesting that thread to access the object; and

when the thread type of the object indicates that it is accessible by any thread, accessing the object from the current thread.

25. The method of claim 24 wherein the thread types include thread-safe and not thread-safe objects.

26. The method of claim 24 including setting a wrapper object to indicate whether the object is accessible only by a certain thread.

27. The method of claim 24 wherein the accessing of the object is an invocation of a member function of the object.

28. The method of claim 27 including mapping invocations of member function from one programming model to another programming model.

29. The method of claim 28 wherein the programming models are the Microsoft Component Object Model and a Java-based programming model.

30. The method of claim 24 including creating the certain thread.

31. The method of claim 24 including setting a wrapper object to contain a reference to the instantiated object, wherein the program indicates the wrapper object when accessing the instantiated object.

32. A computer-readable medium containing instructions for causing a computer system to access objects of varying thread types from a program written in a language that does not support objects of varying thread types, by:

when the program designates to instantiate an object,

determining the thread type of the object;

when the thread type is not thread-safe, sending a message to an appropriate thread to instantiate the object; and

when the thread type is thread-safe, instantiating the object within a currently executing thread; and

when the program designates to invoke a member function of the instantiated object,

when the object is not thread-safe, sending a message to the appropriate thread to invoke the member function of the instantiated object; and

when the object is thread-safe, invoking the member function of the object within the currently executing thread.

33. The computer-readable medium of claim 32 including creating an apartment thread when an apartment-threaded object is to be instantiated.

34. The computer-readable medium of claim 32 wherein the thread type includes a main-threaded object wherein a thread is designated as a main thread, and wherein main-threaded objects are instantiated by the main thread and member functions of main-threaded objects are invoked by the main thread.

35. The computer-readable medium of claim 32 wherein the programming language is a Java-based language and the objects adhere to the Microsoft Component Object Model.

36. The computer-readable medium of claim 32 wherein when the program designates to instantiate an object, a wrapper object is instantiated by the current thread, the wrapper object containing a reference to the instantiated object and an indication of whether the object is thread-safe or not thread-safe.

37. The computer-readable medium of claim 36 wherein the wrapper object contains the identification of an apartment thread when the object is apartment-threaded.

38. The computer-readable medium of claim 36 wherein the wrapper object contains a null identification of a thread when the object is thread-safe, the null identification indicating that the object can be accessed by any thread.

39. The computer-readable medium of claim 32 wherein when an invoked member function returns a reference to an object, storing the identification of the returning thread as the thread to access the referenced object.

40. The computer-readable medium of claim 32 wherein when an invoked member function returns a reference to an object,

saving an indication of the returning thread from which the member function was invoked; and

when invoking a member function of the referenced object,

requesting the returning thread to marshal another reference to the referenced object;

determining whether the marshaled reference is the same as the returned reference;

when the marshaled reference is the same as the returned reference, invoking the member function from the current thread; and

when the marshaled reference is not the same as the returned reference, sending a message to the returning thread to invoke the member function of the referenced object.

41. A computer-readable medium containing instructions for causing a computer system to determine whether a method of an object can be invoked from the current thread, the object being instantiated by an instantiating thread and being identified by a first reference, by:

requesting the instantiating thread to marshal a second reference to the object to the current thread;

when the second reference is received,

determining whether the second reference is the same as the first reference;

when the references are the same, invoking the method of the object from the current thread; and

when the references are not the same, requesting that the instantiating thread invoke the method of the object.

42. The computer-readable medium of claim 41 including saving an indication of whether the instantiating thread should be used to invoke methods of the object so that the instantiating thread is requested to invoke the method of the object for subsequent invocations of methods of the object.

43. The computer-readable medium of claim 41 including saving an indication of whether any current thread can be used to invoke the method of the object so that any current thread can invoke methods of the object.

44. The computer-readable medium of claim 41 wherein the instantiating thread uses a marshaling method of the object to marshal the second reference and wherein when the object is thread-safe, the marshaling member function marshals a pointer that points directly to the object.

45. The computer-readable medium of claim 41 wherein object is developed to adhere to the Microsoft Component Object Model.

46. A computer-readable medium containing instructions for causing a computer system to map access to an object developed with a first programming model to a second programming model, by:

intercepting an attempt to instantiate the object;

instantiating the object and recording an indication that the instantiate object was developed with the first programming model; and

intercepting an attempt to invoke a method of the instantiated object using the recorded indication;

invoking the method of the object; and

mapping parameters returned by the invoked method in accordance with the first programming model to the second programming model.

47. The computer-readable medium of claim 46 wherein the first programming model is the Microsoft Component Object Model.

48. The computer-readable medium of claim 47 wherein the second programming model is a Java programming language model.

49. The computer-readable medium of claim 46 wherein the mapping of parameters includes when a result status that indicates an error is returned by the method, generating an exception.

50. The computer-readable medium of claim 46 wherein the mapping of parameters includes a setting return value of the method to a parameter returned by the method.

51. A computer system for accessing objects of varying thread types in a manner that is transparent to a program accessing the objects, comprising:

an instantiation component that detects when the program indicates to instantiate an object, that requests a certain thread to instantiate the object when the thread type of the object indicates that it is accessible only by the certain thread, that instantiates the object from the current thread when the thread type of the object indicates that it is accessible by any thread, and that sets a wrapper object to contain a reference to the instantiated object; and

an accessing component that detects when the program indicates to access the object using a reference to the wrapper object, that requests a certain thread to access the object when the thread type of the object indicates that it is accessible

only by that certain thread, and that accesses the object from the current thread when the thread type of the object indicates that it is accessible by any thread.

52. The computer system of claim 51 wherein the thread types include thread-safe and not thread-safe objects.

53. The computer system of claim 51 wherein the instantiation component sets the wrapper object to indicate whether the object is accessible only by a certain thread.

54. The computer system of claim 51 wherein the accessing of the object is an invocation of a method of the object.

55. The computer system of claim 54 including mapping invocations of method from one programming model to another programming model.

56. The computer system of claim 55 wherein the programming models are the Microsoft Component Object Model and a Java-based programming model.